Spring 2024

Enhancing Flight Delay Predictions Using Network Centrality Measures

Joseph Ajayi

ENHANCING FLIGHT DELAY PREDICTIONS USING NETWORK CENTRALITY MEASURES

by

JOSEPH AJAYI

(Under the Direction of Yao Xu)

ABSTRACT

Accurate prediction of flight delays remains a formidable challenge within the aviation industry, owing to its inherent complexity and the interconnectivity of its operations. Traditional flight prediction methods frequently utilize meteorological conditions—such as temperature, humidity, and dew point—alongside flight-specific data like departure and arrival times. However, these predictors often fall short of capturing the nuanced dynamics that lead to delays. This thesis introduces network centrality measures as novel predictors for enhancing the binary classification of flight arrival delays. Furthermore, it emphasizes the application of tree-based ensemble models, which are recognized for their superior ability to model complex relationships compared to single-base classifiers. Empirical testing reveals that incorporating centrality measures notably improves the models' average performance. The most effective model achieves an accuracy rate of 86%, surpassing the baseline accuracy by 2%.

INDEX WORDS: Network centrality, machine learning

ENHANCING FLIGHT DELAY PREDICTIONS USING NETWORK CENTRALITY MEASURES

by

Joseph Ajayi

B.S. Troy University, 2020

A Thesis Submitted to the Graduate Faculty of Georgia Southern University
in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ENHANCING FLIGHT DELAY PREDICTIONS USING NETWORK CENTRALITY MEASURES

By

JOSEPH AJAYI

Major Professor:	Dr. Yao Xu
Committee:	Dr. Lixin Li
	Dr. Kai Wang

Electronic Version Approved
May 2024

DEDICATION

I would like to dedicate this thesis to God, my family, and my friends who inspired me throughout my program.

ACKNOWLEDGMENTS

I would like to thank Dr. Xu for her invaluable contributions to this research. I learned a lot while working as her research assistant, and I cannot thank her enough. I would also like to express my gratitude to Dr. Li and Dr. Wang for being part of my thesis committee.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES Page

CHAPTER 1

INTRODUCTION

In the realm of aviation, the efficiency of flight operations significantly hinges on the ability to anticipate and mitigate delays. As the Federal Aviation Administration (FAA) reports, its Air Traffic Organization (ATO) orchestrates the movement of over 45,000 flights daily, servicing 2.9 million passengers across an expansive airspace exceeding 29 million square miles. This volume is projected to swell by 4.9% annually over the next two decades, underscoring a pressing need for robust predictive models that can adeptly forecast flight delays, thereby enabling airlines to optimize scheduling and resource allocation. Despite the proliferation of predictive methodologies ranging from traditional statistical techniques to advanced machine learning algorithms like Decision Trees, Random Forests, Bayesian Networks, and Linear Regression, the quest for high-accuracy predictions remains largely unfulfilled. This challenge is compounded by the unpredictable nature of many delay-inducing factors, such as adverse weather conditions, and the computational demands posed by the voluminous and growing datasets of airline operations.

Against this backdrop, this thesis introduces a novel approach to predicting whether a flight will be delayed or not, leveraging network centrality measures within a binary classification framework. By constructing a network model wherein airports serve as nodes and flight routes as edges, this study integrates centrality metrics to enhance the predictive capabilities of tree-based ensemble models. These models are renowned for their efficacy in capturing complex, non-linear relationships that elude traditional base classifiers. This integration aims to shed light on how the structural properties of the flight network can influence delay propagation and, by extension, overall network performance.

The motivation for this research is twofold: Firstly, flight delays are a pervasive issue that undermines operational efficiency and diminishes passenger satisfaction, with a notable 20% of flights in 2023 experiencing delays across the United States alone. Secondly, existing predictive models often fall short of the accuracy needed for effective planning and resource management, partly due to their reliance on a limited set of predictors that may not fully encapsulate the intricacies of the aviation system. By incorporating network centrality measures into the predictive models, this study aspires to bridge this gap, offering a more comprehensive and nuanced understanding of the factors that contribute to flight delays.

This paper is organized as follows: It begins with a literature review of the current landscape of delay prediction methodologies. Subsequent sections describe the methodology employed in constructing the network model and integrating centrality measures into the ensemble predictive models. The results section presents a comparative analysis of model performances, highlighting the enhanced accuracy achieved through the inclusion of centrality measures. Finally, the conclusion and future works sections reflect on the implications of these findings for airline operations and future research directions in the domain of flight delay prediction.

CHAPTER 2

LITERATURE REVIEW

Gui et al compared random forest and long short-term memory (LSTM) models on a dataset that includes weather, flight schedule, airport information, and automatic dependent surveillance-broadcast (ADS-B) messages. Their tests showed that the LSTM captured the time correlation of flight delay. However, it suffered an overfitting problem due to limited training data. On the other hand, the random forest-based architecture presented better adaptation when handling the limited dataset.

Wei et al created clusters for over 200 Chinese airports with similar network properties and then applied a TS-BiLSTM-Attention model to predict the delay per hour for each airport in the clusters. Since the traditional LSTM is unidirectional, Wei et al combined a forward LSTM layer and a reverse LSTM layer to capture "past" moment information from front to back and "future" moment information from back to front, respectively.

Kim et al used a deep recurrent neural network (RNN) architecture to predict departure delays in two stages. In the first stage, the flight delay status is predicted using the deep RNN. In the second stage, delays of individual flights are predicted using results from the first stage, historical on-time performance, and weather data. The researchers achieved a 90% accuracy for day-to-day delay status prediction. They also discovered that deep input-to-hidden architectures slightly improved their model accuracy.

Güvercin et al proposed a “Clustered Airport Modeling”. In the first step of the approach, the researchers built a network of the airports and extracted graph-based features, including hub score, betweenness centrality, articulation point, in-degree, and weighted-in-degree. Next, they cluster the airports via K-means using the features and delay time series patterns of airports.

Finally, they applied a time series model, REG-ARIMA, to each cluster of airports using the extracted features as regressors. The study showed that betweenness centrality was effective both for clustering the airports and as a regressor in the REG-ARIMA model.

Esmaeilzadeh et al employed a support vector machine (SVM) model to explore the non-linear relationship between flight delay outcomes. Individual flight data were gathered from 20 days in 2018 to investigate causes and patterns of air traffic delays at three major New York City airports. The study revealed that factors such as pushback delay, taxi-out delay, ground delay program, and demand-capacity imbalance were significantly associated with flight departure delay.

Nigam et al used a logistic regression model to predict delay in departure times of aircraft. In addition to airport data, the researchers used weather data such as temperature, humidity, precipitation, and dew point as features for training the model on Microsoft Azure Learning Studio. Their method achieved an 80% accuracy in predicting whether or not a flight would be delayed.

Yu et al proposed a combined Deep Belief Network and Support Vector Regression (DBN–SVR) model, an unsupervised learning method that is combined with a supervised learning algorithm to perform predictive analyses. In the DBN–SVR prediction model, the DBN extracts the main factors with tangible impacts on flight delays, reduces the dimension of inputs, and eliminates redundant information. The output of DBN is then used as the input of the SVR model to capture the key influential factors (leading to flight delays) and generate the prediction value of delays. SVR is used at the top layer to perform supervised fine-tuning within the predictive architecture. The authors examined the performance of the DBN-SVR based on accuracy, robustness, and parameter tuning. Based on the evaluation metrics, the model

successfully outperformed three traditional models: K-Nearest Neighbors, Linear Regression, and Support Vector Machines.

Cai et al proposed a novel model called the Multiscale Spatial-Temporal Adaptive Graph Convolutional Neural Network (MSTAGCN). The researchers first converted the flight delay prediction problem to a time-series analysis task on a time-evolving airport network graph convolutional neural network (GCN). Their model then integrates two advanced multiscale spatial-temporal adaptive graph convolutional layers, each reinforced with a residual connection to enhance the stability throughout the training process. These layers are ingeniously composed of a temporal convolutional block to capture the dynamic temporal patterns of flight delays, and an adaptive spatial convolutional block tailored to model the complex spatial interactions within airport networks. The core of MSTAGCN operates on sequential graph snapshots representing the evolving airport network over time, alongside historical data on flight delays. By processing this information through its dual graph convolutional layers and concluding with a fully connected output layer, the model adeptly forecasts future delays. The performance of MSTAGCN is meticulously assessed through the lens of Mean Squared Error, a statistical measure that quantifies the variance between predicted and actual delay times. Delving deeper, the temporal convolutional block employs the relational GCN (R-GCN) framework to model the spatial dynamics within each snapshot of the airport network, acknowledging both incoming and outgoing flight relations. This approach is further refined to accommodate the temporal evolution of flight delays, leveraging the Markov property for enhanced predictive accuracy. On the spatial front, the adaptive graph convolutional block is introduced to capture unknown spatial interactions, potentially overlooked in conventional models. This block comprises a spatial GCN, a temporal GCN, a dropout layer, and a residual connection, all aimed at accurately modeling the

complex, ever-changing spatial relationships within airport networks. Notably, the model innovates by integrating importance and similarity matrices, which adaptively update to reflect the dynamic significance and connections between airports, thereby capturing emergency and unforeseen flight routes.

Choi et al applied decision trees, random forests, AdaBoost, and K-nearest neighbors to predict weather-induced airline delays. The researchers used specific weather fields such as wind direction angle, wind speed rate, visibility, and precipitation, in addition to flight information such as departure and arrival time, as features for model training. Furthermore, they evaluated the models' performances using a 10-fold cross-validation and Receiver Operating Characteristic (ROC) curve. To balance the data, they employed a Synthetic Minority Oversampling Technique (SMOTE) – an over-sampling approach – to create synthetic minority class examples. The proposed methodology produced an average accuracy of ~75% with sampling and 82% without sampling.

Jiang et al also applied traditional machine learning methods like Support Vector Machines, Decision Trees, Random Forests, and Multilayer Perceptron to the flight delay prediction problem. This is similar to the methodology used by Choi et al. However, Jiang et al went further to apply a Convolutional Neural Network (CNN), which is significant since CNNs are typically used for image recognition tasks. The proposed methodology involves treating the features as a pattern and rearranging their shape as a map before being fed to the network. In the CNN architecture, dense connections between convolutional layers are used along with 1x1 convolution kernels. The researchers also employed Cross entropy as the loss function and a Parametric Rectified Linear Unit (PReLU) as the activation function. The CNN showed a slight increase in accuracy compared to the traditional models.

CHAPTER 3

METHODOLOGY

This study employs a multifaceted methodology integrating network centrality metrics with advanced machine learning models. Initially, the study quantifies the structural importance of airports within the aviation network using centrality measures such as degree, betweenness, and closeness centrality. These metrics highlight key airports that significantly influence flight delay propagation. Subsequently, leveraging Random Forests, Gradient Boosting, and CatBoost algorithms, the research models the complex, non-linear relationships between network characteristics and flight delays. Random Forests mitigate overfitting through ensemble learning, Gradient Boosting sequentially minimizes prediction error by combining weak learners, and CatBoost enhances performance by addressing prediction shift and target leakage.

3.1 Network Centrality

Network centrality is a measure used in network analysis to identify the importance of nodes (individual entities or points) within a network. It provides insights into the structural and functional significance of nodes based on their connections or interactions with other nodes in the network. Network theory is a part of graph theory in mathematics and has extensive applications in various fields including computer science, biology, and social science. A network (or graph) consists of nodes (or vertices) and edges (or links) that connect these nodes.

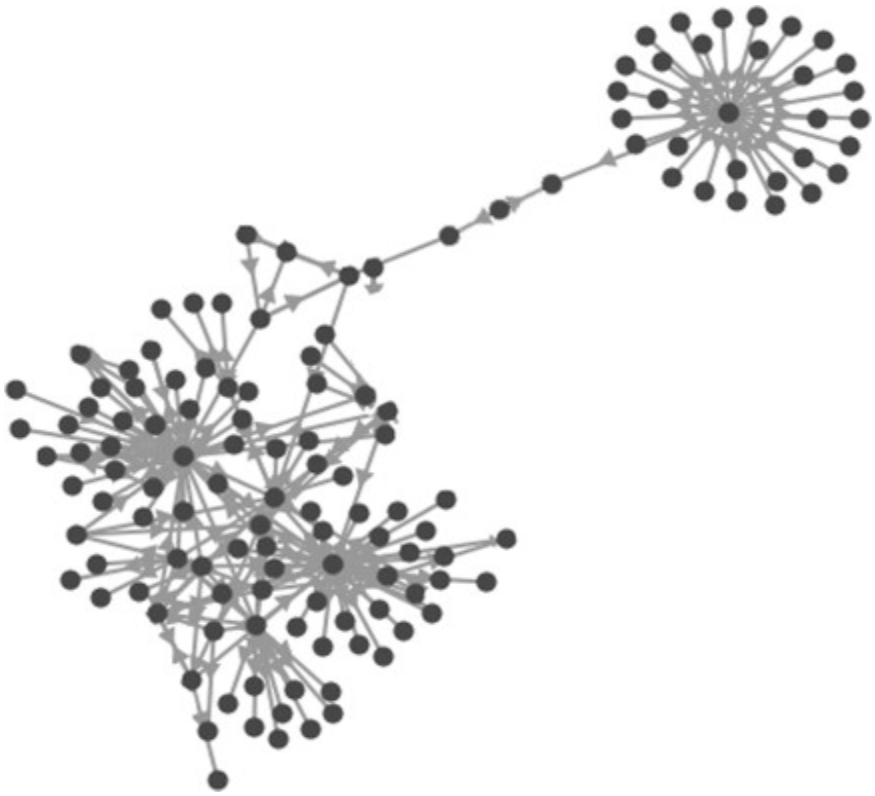

Figure 1: Network diagram

In the context of aviation, nodes can represent airports, while edges can represent flight routes between these airports. There are several measures of centrality, and each provides a different perspective on node importance.

3.1.1 Degree Centrality

This is the simplest centrality measure and is defined as the number of edges incident upon a node. Said another way, the degree centrality of a node is simply its degree—the number of edges it has. The higher the degree, the more central the node is (Golbeck #25). In a directed graph, the degree of a node can be further split into two categories: indegree and outdegree. The in-degree represents the number of edges directed to the node while the outdegree is the number

of vertices that the node directs to other nodes. In a weighted network, the degree centrality of a node is calculated as the sum of weights assigned to the node's direct connections.

In the context of an aviation network, an airport with a high degree centrality would typically be a major hub with a large number of direct flights.

3.1.2 Betweenness Centrality

The betweenness centrality of a node quantifies the number of times a node acts as a bridge along the shortest path between two other nodes. This is measured with the number of shortest paths (between any couple of nodes in the graphs) that pass through the target node (denoted by z). (Layton and Watters #103).

$$\Sigma \frac{\delta x,y(z)}{\delta_{x,y}}$$

$\delta_{x,y}(z)$ is the number of shortest paths that pass through node z while $\delta_{x,y}$ represents the total number of shortest paths in the network. Nodes with high betweenness hold substantial influence within a network because they govern the transmission of information among other nodes. Additionally, the removal of these nodes can greatly disrupt the network's communication, as they are located on the most frequented paths for message exchange

An airport with high betweenness centrality would be one that frequently appears on the shortest routes between pairs of other airports, serving as a critical connector within the network.

3.1.3 Closeness Centrality

Closeness centrality indicates how close a node is to other nodes in a network. Specifically, it is the inverse of the average shortest distance between the vertex and all other vertices in the network. In an airport network, closeness centrality is a measure of the average shortest distance from each airport to each other airports. Airports with high closeness centrality

are often critical for network robustness, as their removal could significantly increase the average distance between nodes, thereby slowing down the movement of passengers and goods. Conversely, airports with lower closeness centrality might be more peripheral, serving fewer destinations directly and relying more heavily on connections through more central airports.

3.1.4 Eigenvector Centrality

Eigenvector Centrality is used to measure a node's influence in a network. In essence, a node with links from important nodes (measured by degree centrality) would have a higher eigenvector centrality than a node with unimportant nodes linking to it. It is determined by performing a matrix calculation to determine what is called the principal eigenvector using the adjacency matrix.

3.1.5 PageRank Centrality

PageRank is an algorithm that was originally developed by Google founders Larry Page and Sergey Brin to rank pages in search results. The original Google paper describes PageRank as the principal eigenvector of the normalized link matrix of the web. Essentially, the algorithm is a variant of Eigenvector Centrality, with the only difference being that it measures the number of incoming links to a node in addition to the importance of the incoming nodes. PageRank is defined as follows:

We assume page A has pages T1...Tn which point to it (i.e., are citations). The parameter d is a damping factor that can be set between 0 and 1. Also, C(A) is defined as the number of links going out of page A (Brin and Page). The PageRank of a page A is given as follows:

$$PR(A) = (1-d) + d\,(PR(T1)/C(T1) + \ldots + PR(Tn)/C(Tn))$$

3.2 Machine Learning Models

Machine learning (ML) is a branch of artificial intelligence that gives computers the ability to learn patterns from data. ML applies to several domains, including healthcare, telecommunications, recommendation systems, and so on. There are several machine learning models. However, three models will be used for this paper: Random Forests, Gradient Boosting, and CatBoost.

3.2.1 Random Forests

Random forests are a combination of predictors called decision trees. Before diving into what a random forest model is, it is important to explain the idea behind decision trees. Decision trees are supervised learning models that make predictions based on possible outcomes. In other words, decision trees 'decide' what the label of a class is based on rules inferred from the relationship between the features.

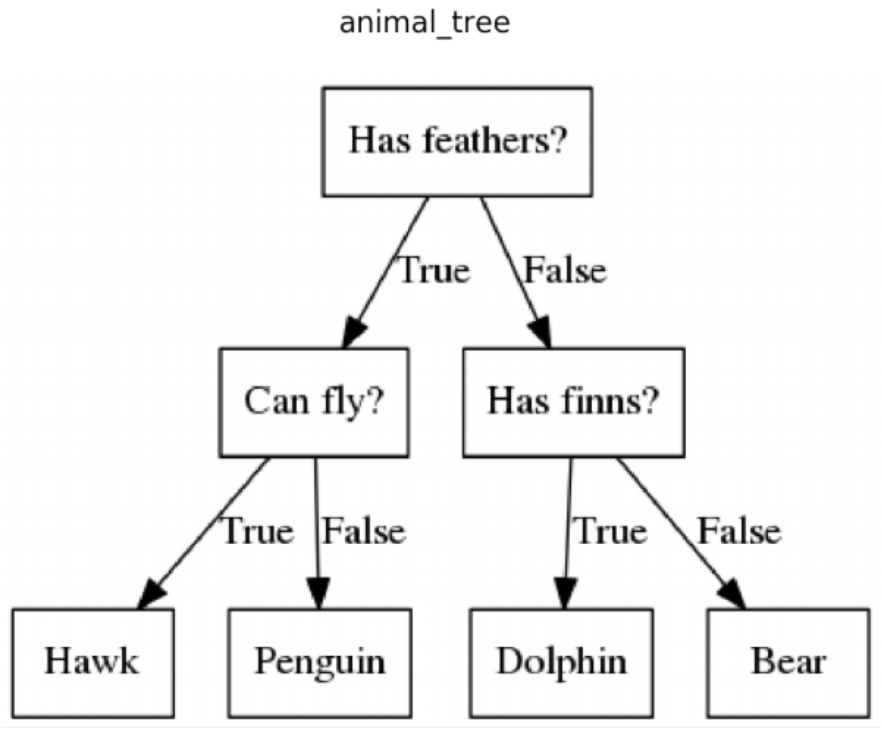

Figure 2: Tree diagram

Because of their structure, decision trees, especially when deep, are prone to overfitting. Hence, a decision tree with a high depth may not generalize well and, consequently, perform poorly on new data. Random forests solve this problem with an ensemble method that involves constructing a set of decision trees from the training data and the combination of the predictions made by the classifiers for class label prediction. The image below shows how it works.

Algorithm 5.5 General procedure for ensemble method.

1: Let D denote the original training data, k denote the number of base classifiers, and T be the test data.
2: **for** $i = 1$ to k **do**
3: Create training set, D_i from D.
4: Build a base classifier C_i from D_i.
5: **end for**
6: **for** each test record $x \in T$ **do**
7: $C^*(x) = Vote(C_1(\mathbf{x}), C_2(\mathbf{x}), \ldots, C_k(\mathbf{x}))$
8: **end for**

Figure 3: The ensemble learning algorithm for Random forests

Random Forests reduce overfitting by manipulating the training data (through bagging) or manipulating input features (through feature randomness). Also known as bootstrap aggregation, bagging is a process in which each tree in the random forest repeatedly samples (with replacement) from the dataset according to a uniform probability distribution, resulting in different trees. This randomization helps to reduce the correlation among decision trees so that the generalization error of the ensemble can be improved. On the other hand, feature randomness is an approach RFs use to randomly select *K* input features to split at each decision tree node. As a result, the decision to split a node is determined from these selected *K* features instead of all the features. After the decision trees are constructed, the ensemble model then takes a majority vote

of all the predictors and returns the result of the tree with the most votes. In addition to classification, random forests can be used for regression analysis which is the focus of this project. Leo Breiman, one of the inventors of random forests, explained that the algorithm works for regression "by growing trees depending on a random vector Θ such that the tree predictor h(x, Θ) takes on numerical values as opposed to class labels."

3.2.2 Gradient Boosting

Gradient Boosting is an ensemble algorithm used in classification and regression tasks. It enhances the predictive accuracy by sequentially combining multiple weak learners into a robust ensemble model. Fundamentally, it operates on the principle that an optimal subsequent model, when synergized with preceding models, minimizes the cumulative prediction error. This process involves iteratively adjusting the target outcomes for the next model to mitigate the error.

To elucidate, the computation of target outcomes for each instance in the dataset is contingent upon the sensitivity of the overall prediction error to modifications in that instance's prediction:

1. If a marginal alteration in an instance's prediction substantially reduces the error, then the ensuing target outcome for that instance is assigned a significant value. Consequently, predictions by the new model that align closely with these targets will diminish the error.
2. Conversely, if a slight change in an instance's prediction does not affect the error, the subsequent target outcome for that instance is zero, indicating that adjusting this prediction will not contribute to error reduction.

The terminology 'gradient boosting' is derived from this methodology, where target outcomes for each instance are determined based on the gradient of the error with respect to the prediction. Each successive model is developed to take a step towards minimizing the prediction error, navigating through the space of potential predictions for each training instance.

Boosting, as an advanced ensemble technique in machine learning, distinguishes itself from traditional models that learn independently from the data. Instead, it amalgamates the outputs of numerous weak learners to formulate a single, substantially accurate strong learner.

Weak Learners Defined:

A 'weak learner' refers to a model that performs marginally better than random guessing. For instance, in classifying mushrooms as edible or inedible, if a model based on random guessing achieves a 40% accuracy, a weak learner would exhibit slightly higher accuracy, ranging between 50% and 60%. Through the integration of several weak learners, boosting aspires to construct a strong learner capable of achieving accuracy levels exceeding 95% for analogous problems.

The decision tree stands out as the preferred choice of weak learners due to its versatility across diverse datasets. For those unfamiliar with decision trees, they are highly recommended for exploration due to their foundational role in many machine learning algorithms.

Gradient Boosting Algorithm: A Detailed Examination

The gradient boosting algorithm is designed for tabular data comprising a set of features (X) and a target variable (y). Its objective mirrors that of other machine learning algorithms: to learn from training data sufficiently to generalize effectively to unseen data points.

Loss Function in Gradient Boosting:

A pivotal component in machine learning, the loss function, enables the quantification of the discrepancy between a model’s predictions and actual values, thus measuring model performance. It serves three critical functions:

- Error Calculation: It compares the model's predicted output against the actual observed values, employing various methodologies to compute the difference.

- Training Guidance: The model strives to minimize the loss function, continually adjusting its parameters to reduce the loss to the lowest possible level.
- Evaluation Metric: By analyzing the loss across training, validation, and test datasets, one can gauge the model's generalization capability and guard against overfitting.

Among the prevalent loss functions, Mean Squared Error (MSE) is widely used in regression tasks within gradient boosting frameworks, emphasizing the importance of accurately measuring and minimizing prediction errors

3.2.3 CatBoost

CatBoost (CB) is a high-performing variant of the gradient boosting (GB) algorithm. CB was proposed by Prokhorenkova1 et al who pointed out that GB suffers from a prediction shift caused by a target leakage. Hence, the researchers proposed an alternative algorithm, called Ordered Boosing, to fix the problem.

Ordered Boosting

Traditional gradient boosting methods can easily overfit the training data. CatBoost addresses this issue through ordered boosting, a novel approach that reduces overfitting without significantly increasing the computational cost. It involves creating a random permutation of the training data and, for each instance, fitting models using only the data before it in the permutation. It is important to note that CatBoost uses different permutations for different steps of gradient boosting. This approach ensures that the model being trained does not see the current data point, thereby reducing the likelihood of overfitting.

Algorithm 1: Ordered boosting

input : $\{(\mathbf{x}_k, y_k)\}_{k=1}^{n}$, I;

$\sigma \leftarrow$ random permutation of $[1, n]$;
$M_i \leftarrow 0$ for $i = 1..n$;
for $t \leftarrow 1$ **to** I **do**
    for $i \leftarrow 1$ **to** n **do**
        $r_i \leftarrow y_i - M_{\sigma(i)-1}(\mathbf{x}_i)$;
    for $i \leftarrow 1$ **to** n **do**
        $\Delta M \leftarrow$
          $LearnModel((\mathbf{x}_j, r_j) :$
          $\sigma(j) \leq i)$;
        $M_i \leftarrow M_i + \Delta M$;
return M_n

Figure 4: Ordered Boosting algorithm

CHAPTER 4

ANALYSIS AND RESULTS

The experiment involved the following steps:

4.1: **Data Collection**

The dataset employed in this research was acquired from the U.S. Bureau of Transportation Statistics (BTS), comprising detailed records of on-time arrival and departure for non-stop domestic flights. Specifically, origin airport ID, destination airport ID, departure time, and arrival time were selected. For this study, data spanning July 2022 to June 2023 were extracted.

4.2: **Data processing**

After the extraction process, the data set was imported using the pandas package in Python. The data set containing the latitudes and longitudes of each airport was merged with the flight data. Next, records containing null or missing values were removed, thereby ensuring the integrity and reliability of the dataset for analytical purposes. Since the goal was a binary classification of a flight's arrival, the arrival delay column was converted to a categorical format by setting parameters greater than zero to 1 and there rest to 0.

Table 1: Attribute description of the dataset

Attribute Name	Description	Type
ORIGIN_AIRPORT_ID	Origin	Integer
DEST_AIRPORT_ID	Destination	Integer
DEP_TIME	Actual departure time	Float
DEP_DELAY	Number of minutes a flight is	Float

	delayed for	
Origin_Degree_Centrality	Origin Degree Centrality	Float
Destination_Degree_Centrality	Destination Degree Centrality	Float
Origin_Betweenness_Centrality	Origin Betweenness Centrality	Float
Destination_Betweenness_Centrality	Destination Betweenness Centrality	Float
Origin_Closeness_Centrality	Origin Closeness Centrality	Float
Destination_Closeness_Centrality	Destination Closeness Centrality	Float

4.3: **Network Graph Construction**

With the data processed, a network graph representing the airport system was created using the NetworkX package in Python. The plot below represents the top 20 airports with the highest scores for each centrality measure.

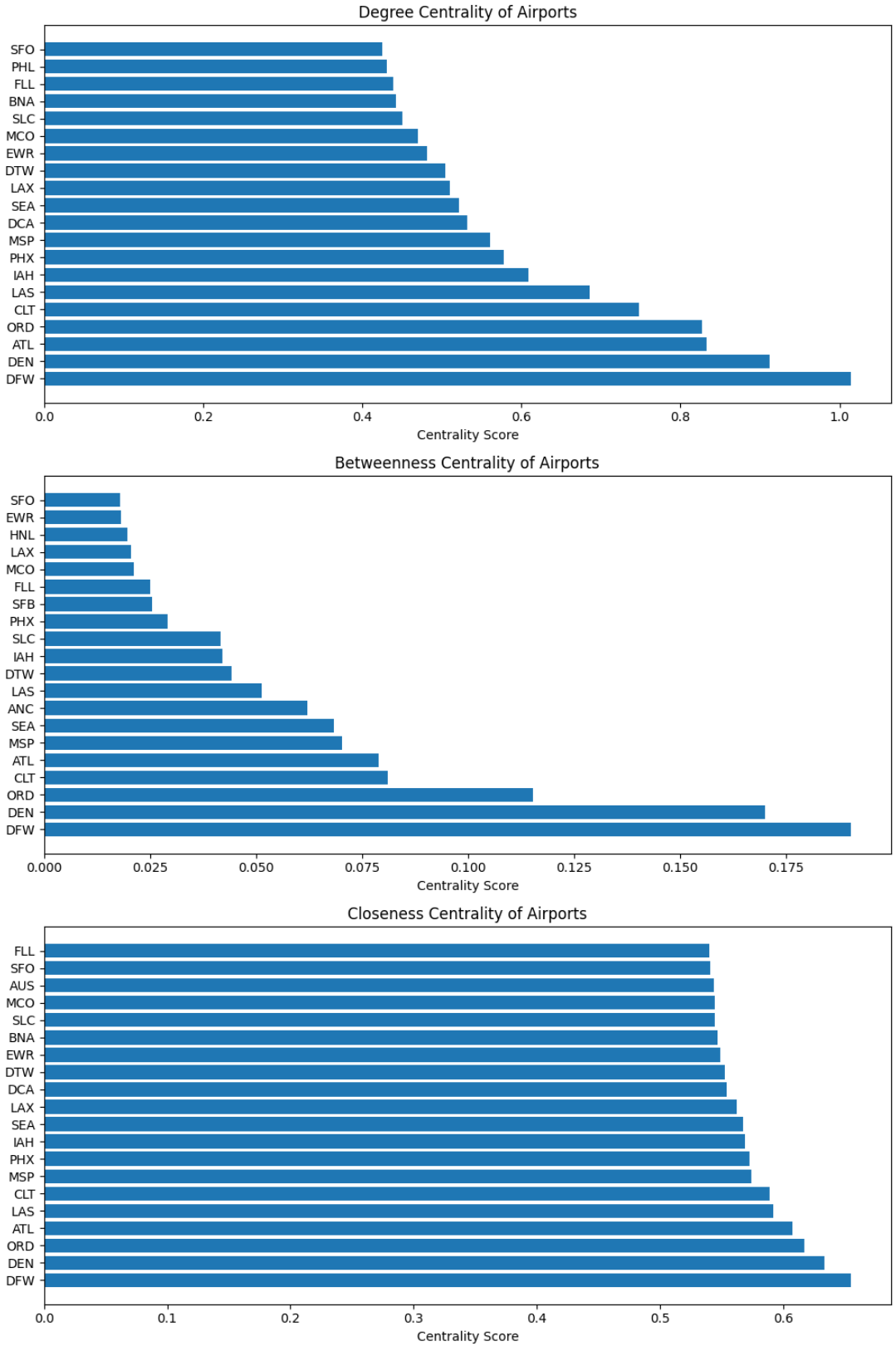

Figure 5: Top 20 centrality scores in the airport network

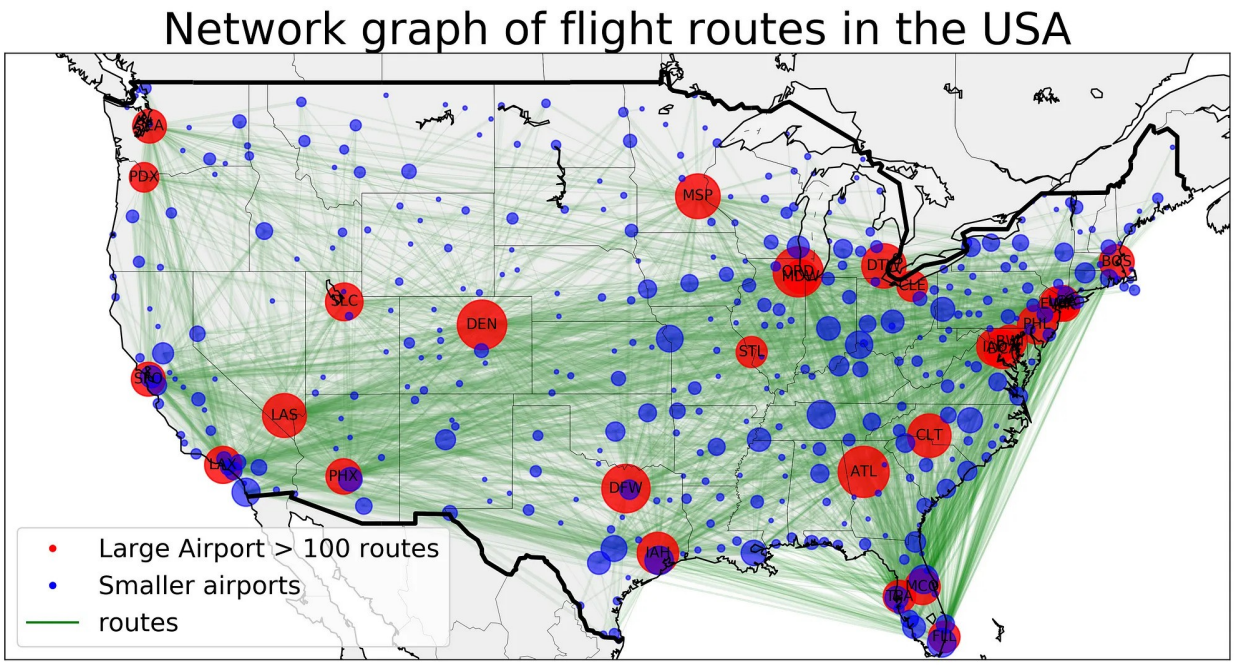

Figure 6: U.S. airport network (Towards Data Science)

Next, the dataset was split into 80% training and 20% testing. To reduce variance and risk of overfitting, the dataset was also shuffled. In this paper, the target variable for prediction was designated as the arrival delay, encompassing both positive (indicating delays) and negative (indicating arrivals on or before time) values. These values were transformed into categorical variables, with positive delays coded as one and on-time or early arrivals coded as zero.

4.4 Model Training

For this study, I used the default parameters in the Random Forest and Gradient Boosting methods provided by Scikit-learn. For CatBoost, I used the open-source algorithm provided by Yandex. The model training was split into two phases. In the first phase, the models underwent training exclusively on a set of baseline features, namely the origin, destination airport, departure delay, and departure time.

4.5 Model Evaluation

To get a better sense of the impact of the centrality measures on prediction performance, the models were evaluated with the following metrics:

4.5.1 Permutation Feature Importance

The permutation feature importance (PFI) is a model inspection technique that is used to measure the contribution of each feature to a model's performance. Unlike intrinsic methods that rely on the specific structure of a given model (e.g., feature importance in decision trees), permutation feature importance is model agnostic, making it applicable across a wide range of model types.

The core idea behind permutation feature importance is straightforward: by randomly shuffling the values of a single feature in the dataset, one can break the association between that feature and the target outcome. If the model's performance deteriorates significantly after this permutation, it indicates that the model relied heavily on that feature for making predictions. Conversely, if the performance remains largely unchanged, the feature is likely not crucial for the model's predictions. PFI can be calculated multiple times with different permutations of the same feature; doing this provides a measure of the variance in the estimated feature importances for the specific trained model. PFI is calculated as follows:

1. Empirical Loss Calculation ($E(f)$):
    - Train the predictive model (f) using the full dataset Z
    - Calculate the original empirical loss $E(f)$, using the model's predictions and the actual outcomes. This is done by averaging the loss L over all instances in the dataset, providing a baseline measure of model performance.

2. Switch Operation:

- For a given feature set *X1*, perform a "switch" operation by systematically recalculating the model's loss after switching *X1* values among all pairs of observations, while keeping *X2* and y paired as originally. This step assesses the impact of reshuffling *X1*'s values on model performance.

3. Divide Operation (alternative method for large datasets):
    - As an alternative to the computationally intensive switch operation, divide the dataset in half and swap *X1* values between these halves. Then, calculate the model's loss for these adjusted pairings. This operation serves as a simpler method to estimate the importance of *X1*.
4. Empirical Model Reliance (MR) Estimation:
    - Calculate the empirical MR as the ratio of the switch operation to $E(f)$, reflecting the model's reliance on the feature set *X1*. A higher MR indicates a greater dependency on *X1* for making accurate predictions.
5. Repeat for Additional Features:
    - If assessing multiple features or feature sets for their importance, repeat steps 2 and 3 (and potentially step 4) for each feature or set of features under consideration.

To analyze each model's reliance on each feature, I implemented a function to calculate the PFI scores, which were then ranked from highest to lowest, as demonstrated in the figures below.

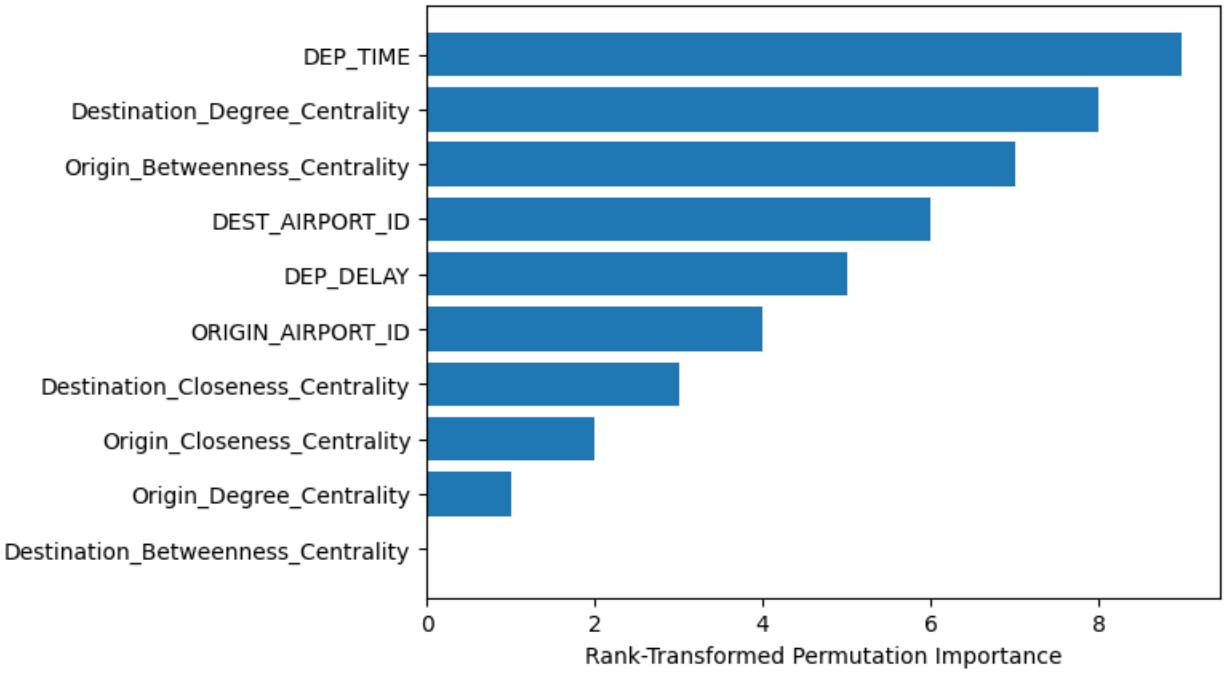

Figure 7: Permutation Feature Importance for Random Forests

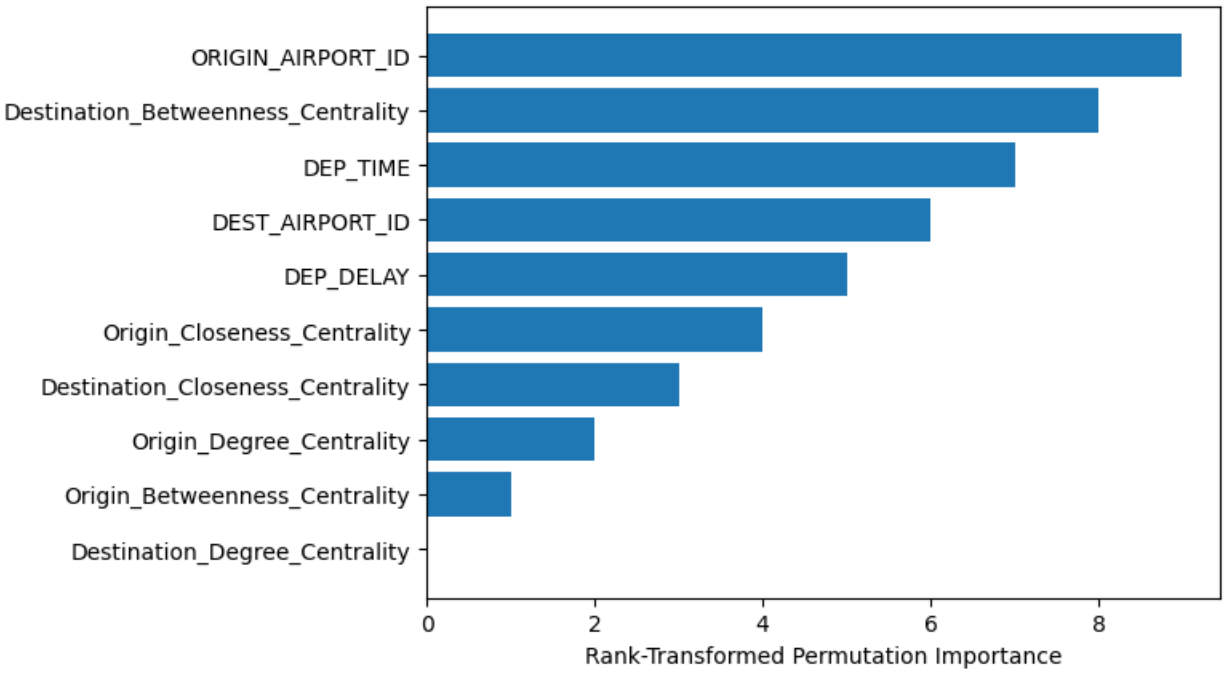

Figure 8: Permutation Feature Importance for Gradient Boosting

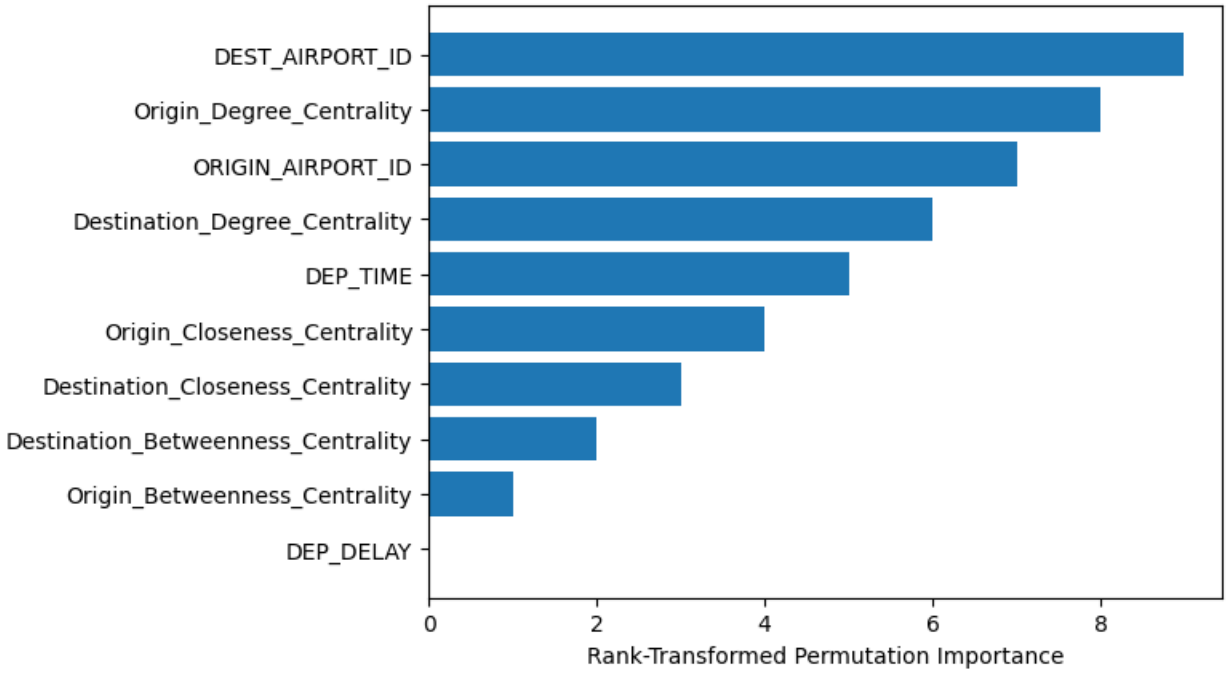

Figure 9: Permutation Feature Importance for CatBoost

4.5.2 Cross Validation

Cross-validation (CV) is a cornerstone technique in the field of machine learning, crucial for assessing the predictive performance of models. This technique is especially valuable in applied machine learning tasks, where the goal is to develop models that generalize well to unseen data. CV's primary advantage lies in its ability to mitigate model evaluation biases and variances, offering a more reliable estimate of model performance across different subsets of data.

At its core, cross-validation involves repeatedly dividing the dataset into distinct training and testing sets, training the model on the former, and evaluating it on the latter. This process not only aids in comparing the effectiveness of different models but also in fine-tuning the parameters of a single model. Unlike simpler validation approaches that split the data only once,

CV systematically uses different portions of the data for training and validation, ensuring that the evaluation is robust and less susceptible to the idiosyncrasies of a single data partition.

4.5.3 Receiver Operating Characteristic (ROC) Curve

The ROC curve is a graph showing the performance of binary classifiers. The plot illustrates the True Positive Rate (TPR) vs False Positive Rate (FPR) at different classification thresholds.

$$\text{TPR} = \frac{TP}{TP + FN}$$

$$\text{FPR} = \frac{FP}{FP + TN}$$

The area under the ROC curve — also known as the AUC curve — from (0, 0) to (1, 1) is used as an accurate measure of prediction performance. The closer the AUC score is to 1, the better the prediction.

Table 2: Metrics for models trained with baseline features

Model	Accuracy	Precision	Recall	F1 Score
Random Forest	0.845	0.863	0.729	0.790
Gradient Boosting	0.851	0.876	0.733	0.798
CatBoost	0.850	0.885	0.720	0.795

Table 3: Metrics for models trained with baseline and centrality-based features

Model	Accuracy	Precision	Recall	F1 Score
Random Forest	0.862	0.895	0.742	0.812
Gradient Boosting	0.858	0.888	0.740	0.808
CatBoost	0.856	0.885	0.735	0.803

The tables above show that the metrics of all the models increased after the centrality values were added as features.

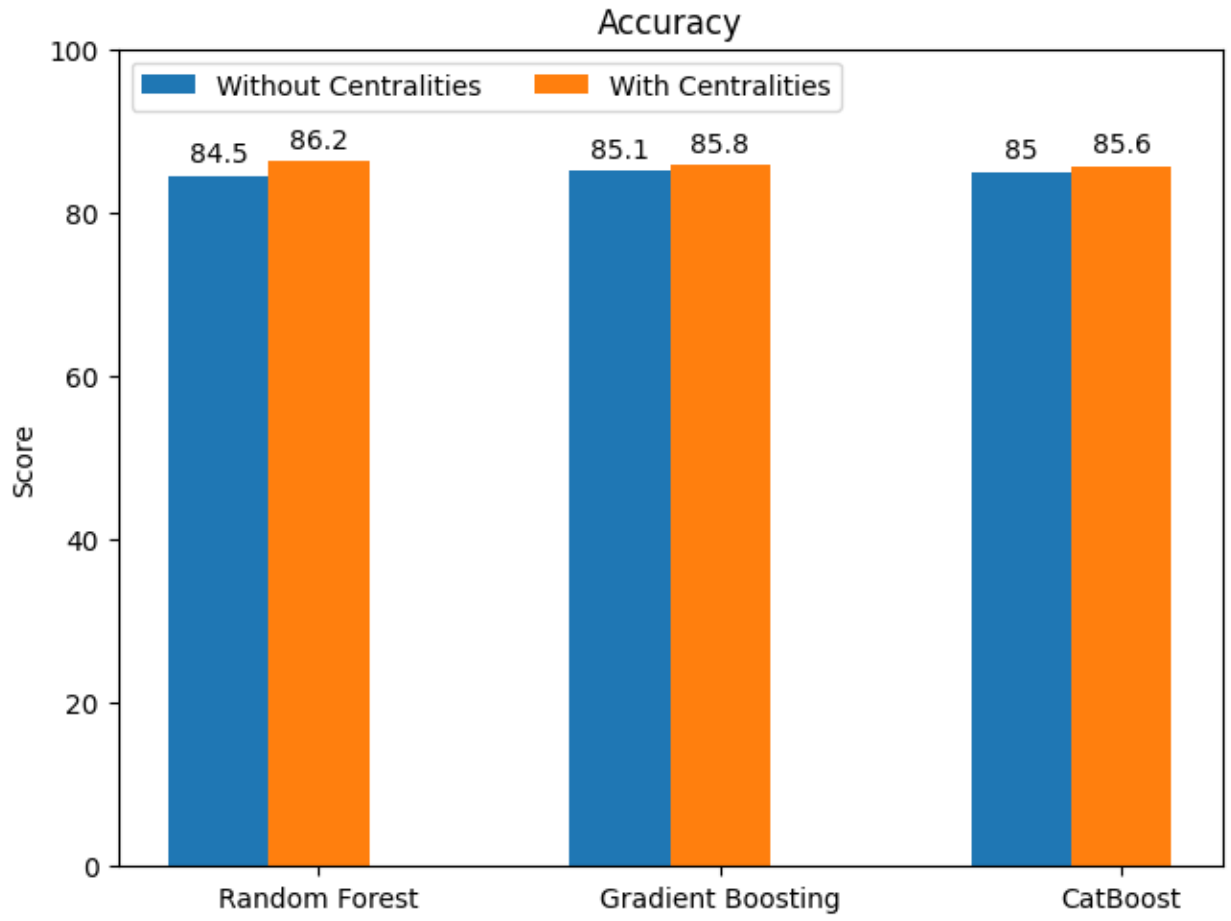

Figure 10: Grouped bar chart of accuracy scores

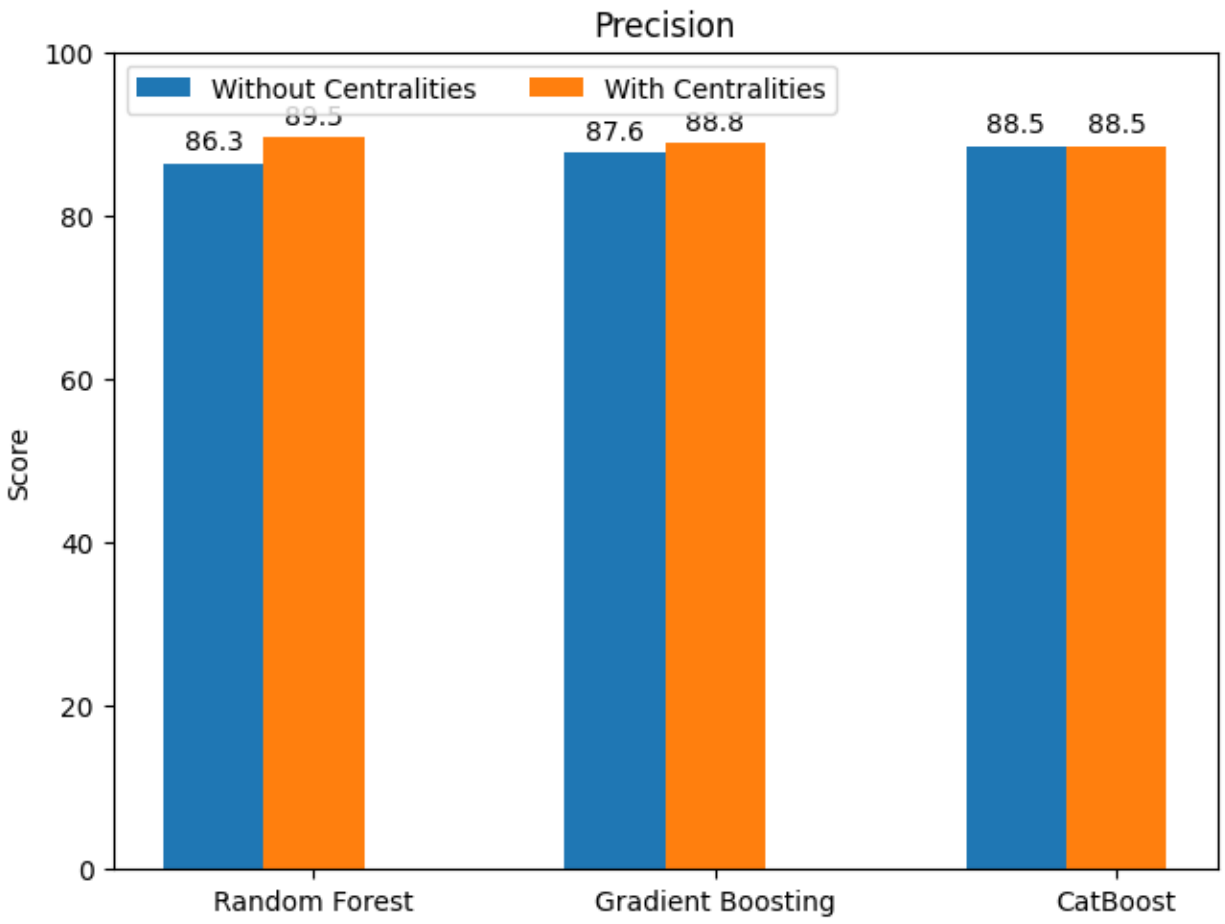

Figure 11: Grouped bar chart of precision scores

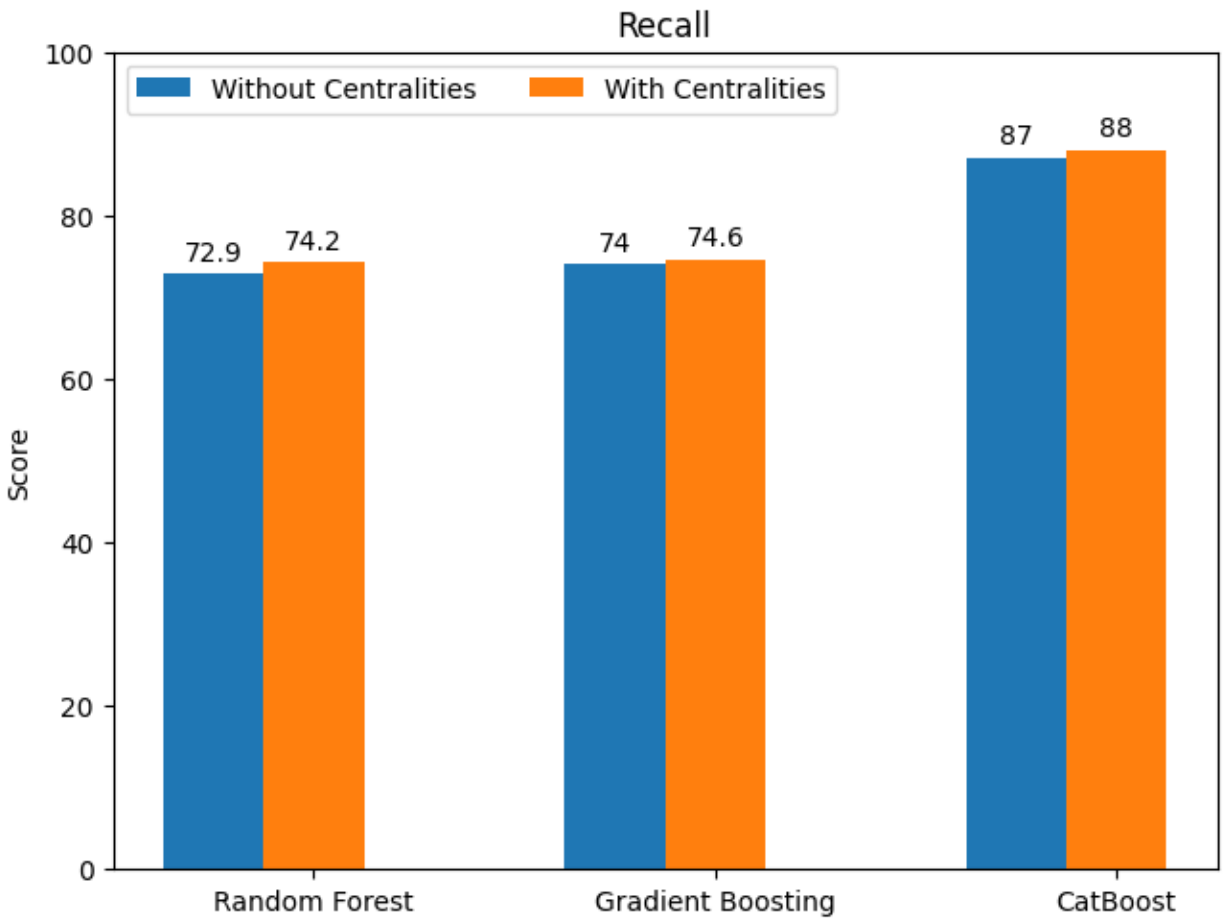

Figure 12: Grouped bar chart of recall scores

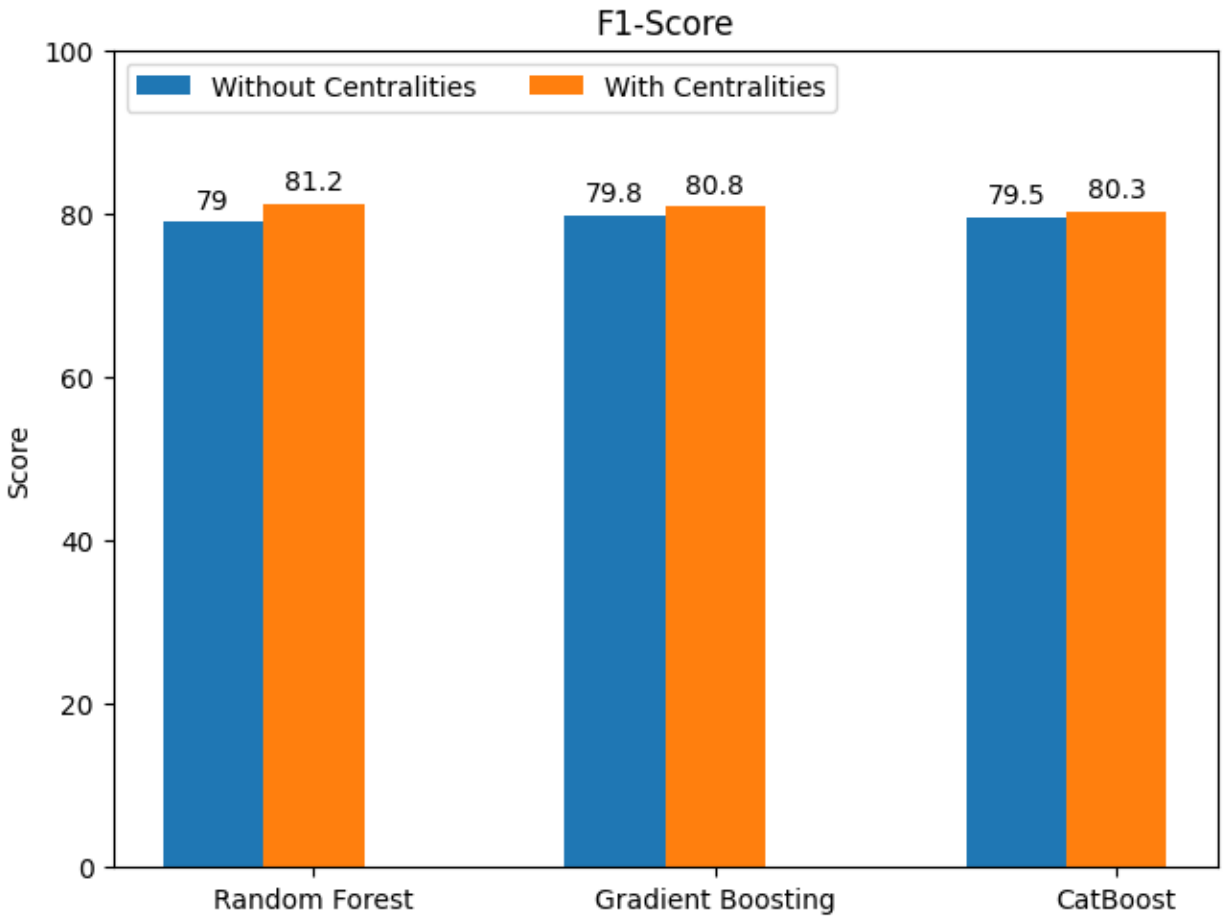

Figure 13: Grouped bar chart of F1 scores

CHAPTER 5

CONCLUSION

In this study, I showed that centrality measures — particularly degree, betweenness, and closeness centrality — can improve the performance of traditional machine learning models for predicting flight delays. In addition, the goal was to prove that this methodology would work well in the real world. By using several evaluation metrics like recall, F1 score, accuracy, and precision, I compared the models trained with centrality and traditional features to the models trained with only the traditional features. My tests show that the centrality measures increased the evaluation metrics of the models.

Furthermore, I used the PFI to calculate the importance of each feature and also to show that the models relied on the centrality measures in making predictions. This study would be valuable for government agencies like the Federal Aviation Administration (FAA) that oversee the aviation industry in the United States.

CHAPTER 6

FUTURE WORK

In this study, I explored the integration of centrality measures alongside traditional features like destination airport ID, origin airport ID, departure time, and departure delay to enhance predictive models in the context of air traffic networks. This innovative approach opens up several avenues for future research to further understand and leverage the complex dynamics of air traffic systems.

One promising direction for future work involves a detailed comparative analysis of various centrality measures to assess their predictive power across different scenarios within air traffic management. By examining how different centrality metrics influence predictions, researchers can identify which measures are most relevant for specific predictive tasks, such as forecasting flight delays or determining route popularity.

The concept of treating air traffic networks as multilayer networks presents another intriguing research opportunity. In such networks, layers could represent various airlines, flight types (e.g., domestic versus international), or time slots, enabling a more granular analysis of network dynamics and their implications for air traffic predictions. Furthermore, extending the analysis to include additional network features—beyond centrality—such as community structure, network resilience, or network efficiency, could provide deeper insights into the factors influencing air traffic systems. This broader perspective might reveal new strategies for optimizing flight scheduling, route planning, and congestion management.

Exploring the operational impact of integrating centrality measures into predictive models also warrants further investigation. Collaborating with industry stakeholders, such as airlines and airport authorities, could help assess the practical benefits and challenges of

implementing these advanced models in real-world settings. Such collaborations could also facilitate studies on the scalability of these models, particularly for applications requiring real-time or near-real-time predictions, which are crucial for effective air traffic management.

Moreover, the potential of this research extends beyond the immediate realm of air traffic management. Investigating whether the findings from air traffic networks can be applied to other types of transportation networks could uncover universal principles applicable to broader transportation system optimization efforts. Additionally, examining the environmental and economic impacts of optimizing air traffic operations based on network-centric predictive models could contribute valuable insights into achieving sustainable and economically viable transportation solutions.

REFERENCES

G. Gui, F. Liu, J. Sun, J. Yang, Z. Zhou and D. Zhao, "Flight Delay Prediction Based on Aviation Big Data and Machine Learning," in IEEE Transactions on Vehicular Technology, vol. 69, no. 1, pp. 140-150, Jan. 2020, doi: 10.1109/TVT.2019.2954094.

Wei, X.; Li, Y.; Shang, R.; Ruan, C.; Xing, J. Airport Cluster Delay Prediction Based on TS-BiLSTM-Attention. Aerospace 2023, 10, 580. https://doi.org/10.3390/aerospace10070580

Y. J. Kim, S. Choi, S. Briceno, and D. Mavris, "A deep learning approach to flight delay prediction," 2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC), Sacramento, CA, USA, 2016, pp. 1-6, doi: 10.1109/DASC.2016.7778092.

Esmaeilzadeh, E., & Mokhtarimousavi, S. (2020). Machine Learning Approach for Flight Departure Delay Prediction and Analysis. Transportation Research Record, 2674(8), 145-159. https://doi.org/10.1177/0361198120930014

R. Nigam and K. Govinda, "Cloud based flight delay prediction using logistic regression," 2017 International Conference on Intelligent Sustainable Systems (ICISS), Palladam, India, 2017, pp. 662-667, doi: 10.1109/ISS1.2017.8389254.

Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. In Proceedings of the seventh International Conference on World Wide Web (pp. 107-117). Elsevier Science Publishers B. V.

Bin Yu, Zhen Guo, Sobhan Asian, Huaizhu Wang, Gang Chen. "Flight delay prediction for commercial air transport: A deep learning approach." Transportation Research Part E: Logistics and Transportation Review, Volume 125, 2019, Pages 203-221, ISSN 1366-5545, https://doi.org/10.1016/j.tre.2019.03.013.

K. Cai, Y. Li, Y. -P. Fang and Y. Zhu, "A Deep Learning Approach for Flight Delay Prediction Through Time-Evolving Graphs," in IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 8, pp. 11397-11407, Aug. 2022, doi: 10.1109/TITS.2021.3103502.

S. Choi, Y. J. Kim, S. Briceno, and D. Mavris, "Prediction of weather-induced airline delays based on machine learning algorithms," 2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC), Sacramento, CA, USA, 2016, pp. 1-6, doi: 10.1109/DASC.2016.7777956.

Liudmila Prokhorenkova , Gleb Gusev, Aleksandr Vorobev , Anna Veronika Dorogush, Andrey Gulin1 (2018). CatBoost: gradient boosting with categorical features support